APPLICATION OF LINEAR GRAPHS TO COMMUNICATION PROBLEMS

by
Arthur E. Laemmel

Research Report R-353-53, PIB-287
Office Of Naval Research
Contract NOnr-839(05), Proj. No. 075-216

December 11, 1953

MRI

POLYTECHNIC INSTITUTE OF BROOKLYN
MICROWAVE RESEARCH INSTITUTE

ACKNOWLEDGEMENT

The research described in this paper was supported by the Office of Naval Research under Contract NOnr-839(05), Project Designation NR-075-216.

ABSTRACT

Codes have been discussed recently in connection with signal compression and noise reduction in communication theory, and they are here defined as transformations between two time series of discrete symbols. The electronic apparatus which performs the transformation, or any part of the apparatus, can be described by a linear graph which represents transitions between different memory-states. It will be shown how these graphs can be used to systematically classify codes, combine different blocks of apparatus, examine synchronism of decoding apparatus, etc. Linear graphs have been studied as mathematical systems in connection with topology, Markov chains, and electrical network theory; hence there is already much theory which might be profitably applied to communication problems.

Application of Linear Graphs to Communication Problems

State diagrams

It is the purpose of this paper to present a method of easily visualizing certain features of the operation of finite-state transducers. A finite-state transducer is a piece of apparatus with one or more inputs and one or more outputs, all bearing digital signals with symbols chosen from a finite alphabet. Such symbols, for example, might be the binary digits 0 and 1, or the letters of the common alphabet. The symbols in these signals occur in time sequence, but not necessarily periodically. Suppose now that a transducer has one input and one output, and that the input symbols are fed in periodically. Every time an input symbol is fed into the transducer either of three things may be observed at the output: 1) nothing comes out, 2) a single symbol comes out, or 3) a finite group of symbols comes out rapidly but in a certain sequence. When an output symbol, or group of symbols, does come out it is a function not only of the present input symbols, but also of previous input symbols. The device might be said to have a certain memory of previous symbols. It can only remember a finite number of things about the previous symbols, but this is not to say that there is any fixed distance into the past from beyond which there is no influence. Memory of a only finite number of things about past symbols is implied by the fact that the apparatus is to be physically realizable, is of finite extent and accuracy, and therefore is capable of assuming only a finite number of recognizable memory states. As an example, consider a device which has three flip-flops and a magnetic drum storage with 100 magnetizable elements. A particular memory state might correspond to two flip-flops "off", the other "on", and a certain pattern of 20 elements magnetized on the drum. The total number of memory states in this example is $2^3 \cdot 2^{100} \simeq 10^{31}$. Since in many cases the number of states will be correspondingly large, it is evident that the diagram to be described below cannot then be drawn in full; but usually a simpler case of the same type will suffice, or else certain features of the full diagram can be discussed without actually drawing it in full. The class of finite-state transducers was introduced by C.E. Shannon(1) in connection with coding operations.

The operation of finite-state transducers can be easily visulized by using a linear graph diagram. Fig. 1 shows how the linear graph representation compares with three others for a specific type of code: 1) the upper right shows the code table, 2) the lower table shows how a particular sequence is transformed, and 3) the heavy lines in the diagram show a "tree" which represents the successive binary decisions necessary to decide what the input combination is, and then what the corresponding output is. As each input combination is completed the next must be begun, therefore all of the terminal branches are returned to the base state, A.

---

(1) C.E. Shannon, "A Mathematical Theory of Communication", Bell System Tech. Journ., Vol 27, pp 379-423, July 1948 - see Section 8

The meanings of the two states in this example are: state A is the state the apparatus is in if an input combination has just been completed and the next is about to begin, state B is the state of knowing that the input combination has started with 0 and waiting to see if the whole combination is 00 or 01. The bottom table in Fig. 1 shows one input sequence but two output sequences, these represent the outputs which occur respectively when the apparatus is started in state A and in state B. Thus, one transducer with σ states really can effect σ possible transformations of a semi-infinite sequence, and the linear graph represents all of these at once. A linear graph with labels such as in Fig. 1 will be called a state diagram if it fulfills the following conditions: 1) has n arrows leaving each node (state), each labeled with one of the n symbols in the alphabet, and each terminating in the same or another state, and 2) has another label, in parenthesis, representing the output symbols, if such occur during that particular change of state. Several operations on such state diagrams will now be defined, after which some applications to practical problems will be given.

Addition

In Fig 2. two different transducers are shown connected so that they have a common input. If one transducer can have σ_1 states and the other σ_2 states, then when combined they can have $\sigma_1\sigma_2$ states. The sum of two state diagrams, each representing a transducer, will be defined as another state diagram which represents the two transducers with a common input. If the states of the first transducer are A and B and of the second α and β, then the states of the combined transducers can be represented as, $A\alpha$, $A\beta$, $B\alpha$ and $B\beta$. The transition arrows for the sum diagram are obtained by noting the transitions which take place in the devices individually. For example, suppose the first device is in state A, the second device is in state β, and a symbol 0 is fed in. Then the first device will shift to state B, the second device will remain in state β, therefore the combined devices will shift from state $A\beta$ to $B\beta$. By repeating this process the sum diagram can be completed as that n(2^2 in this case) arrows leave each state. The outputs have been omitted, if they are desired then two outputs must be shown in the sum graph, and this might be done by writing one above the other in parentheses on the proper arrow. (See section "Networks of Transducers" below.) The operation of addition defined here, or that of multiplication defined below, do not agree with the existing operations on linear graphs.(2)

Reduction

In Fig 3 is shown a state diagram with two dashed lines thru it. These dashed lines have the following property: if a state on one side of the line is occupied an input symbol will either cause a crossing of the line or a staying on the same side of the line, independently of just which state on that side is initially occupied. A state diagram which describes only the

(2) D. König, "Theorie der endlichen und unendlichen Graphen," Chelsea Pub. Co., New York 1950

crossing of such a line, or more generally between any sets of subsets of the original states, will be defined as a reduction of the whole state diagram. Thus in Fig. 3 the diagram with A and B + C represents the crossings of the horizontal line. Note that when A_β was written in a state it corresponded to A and β, here B + C means B or C.

Factorization

A state diagram can sometimes be reduced in several ways; if it can be reduced in enough ways so that the totality of the reduced states gives enough information to determine the original state, than it can be factored as shown in Fig. 3. Factorization is almost the inverse operation to addition, except that if the factors are readded (according to the first definition of addition) more states might appear which were not present in the original. For example, the factors in Fig 3 are the same as the addends in Fig. 2, but the state B_α in Fig. 2 is not represented in Fig. 3. However, the state B_α is quitted as the first input is received, and hence it does not affect the steady-state operation of the device.

Multiplication

The multiplication of two state diagram is defined to correspond to the connection of two transducers in tandem. Agan the number of states in the product is the product of the numbers of states in the original diagrams. The output of the first transducer becomes the input to the second. Thus in Fig 4, 1011 into the first transducer becomes 0111, this is fed to the second transducer and comes out 0101111 (for a starting state A_α). Note that the same result is obtained in one step by using the product diagram. The product diagram is built up as follows: take a pair of states, say B_α, and an input symbol, say 0. The first transducer changes to state A and sends a 01 to the second; the second then stays in state α emitting 0, and then changes to state β emitting 10. The combination in tandem thus goes from state B_α to state A_β with the emission of 010, and an arrow is so drawn in the product state diagram. Repetition of this process 7 more times completes the product graph.

Identities

An "identity" will be defined here to mean a state diagram which, if it is started in the correct state, will transform any semi-infinite (input) sequence, after a possible delay into the same (output) sequence. In Fig 5 the upper diagram represents a transducer which always immediately emits the same symbol which is fed in. The middle diagram represents a transducer which always emits an output symbol the same as the immediately preceding input symbol. By continuing this series the state diagram for any delay line can be obtained; for a delay of δ symbols 2^δ states are required. Another type of identity is shown in the lower diagram of Fig. 5, here if a start is made

in the left hand state a 1 will come out as a 1 immediately, a 01 will come out as a 01 but all at the time of the second digit, and a 000 will come out as 000 all at the time of the third digit. Thus any sequence is transformed into itself, but the delay is not a fixed amount.

Inverse

An inverse of a state diagram will be defined as any other state diagram which can be multiplied by the first to yield an identity.* Since there is more than one identity the inverse is not unique, in fact another inverse can always be found by multiplying a particular inverse by an identity. The inverse diagram of T corresponds to a transducer which performs a decoding opertion which restores the original signal after an encoding corresponding to T. Such a situation is shown in Fig. 6; the inverse in this examples happens to be the same as the original state diagram. If the encoder is started in state A and the decoder in state α, the combination is started in state $A\alpha$ and it will be seen that any semi-infinite sequence will be transformed into itself with no error. If however a start is made in any other combination of states the first few digits will be wrong, but eventually no more errors will be made. Thus, starting in state A_β and with an input sequence 00101100..., the output sequence will be 01101100....

Networks of transducers

The operations of addition and multiplication can be used to derive a single state diagram for a network of transducers. In Fig. 7 A and B are the state diagrams for a flip-flop and one symbol delay, and A + B is the diagram of these two units fed by a common input. A coincidence circuit is represented by the diagram C, which has only a single state. Note that A + B has one input and two outputs, while C has two inputs and one output. The combination (A + B) x C then represents the interconnection of all three units as shown.

If the network to be analyzed involves feedback the above method can still be applied with a slight modification. Fig 8 shows a network with a coincidence circuit which has one input fed by an external source and the other fed by the output delayed one symbol. The first step is to find the state diagram of the same network with the feedback loop opened. As can be seen, this network has two inputs and two outputs. By an operation which might be called contraction, only those arrows and labels in which the lower input and lower output agree are retained, the labels then being the corresponding upper parts. This will always be possible since there are arrows leaving every state labeled with every input combination. The physical significance of this operation is that the second input to the coincidence unit is the output of the delay unit, hence only combinations in which these are the same are possible, and when they are the same they need not be shown

* Since multiplication is not commutative, the right and left inverses will in general differ. The right inverse will be meant below unless otherwise specified.

since the desired diagram describes only the external behavior of the network.

Synchronization

Any transducer which has a memory of previous input symbols must be started in the correct state to accomplish a given transformation of the signal. In Fig. 1 the transducer codes according to the table only if it is started in state A, but if it is started in state B instead, the lower table shows that the desired code is eventually reached, at least for the input sequence shown. This situation might be described by saying that the transducer starts out of sync but later gets into sync by itself. Testing a transducer by starting it in a wrong state and feeding in different sequence to see if it corrects itself would be a tedious process, but by simply adding the diagram to itself (according to the above definition of addition) shows at a glance from which states sync may eventually be regained and which input sequence will accomplish this. A diagram added to itself represents two similar devices fed by the same signal, and also represents the transitions of state-pairs describing the actual transducer which may be out of sync, and an ideal transducer which is in sync. For example state A_β might mean mean that the actual device is in state A but it should be in state B, and A_α would then mean in state A when it should be in state A, i.e. in sync. The state diagram shown in Fig 1 is shown added to itself in Fig. 9. It can now be seen at a glance that out of sync operation will be maintained only if a long sequence of 0's is fed in. The first 1 returns the device to the $A_\alpha - B_\beta$ part of the linear graph, and this is identical with the original state diagram. A device of this type is self resetting.

An example of a non-self resetting device is also shown in Fig. 9. In this case when the diagram is added to itself a linear graph with two separate parts is formed. There is no path from state A_β to state A_α, therefore if the device is set in state A when it should be in state B the error will never be corrected in normal operation. In this case the diagrams are not needed to see that sync will not be restored since it is obvious that 0 1 0 1 0 ... might be interpreted either as 01 01 0... or as 0 10 10..., and since all messages are the same length the mistaken decomposition will be perpetuated. In more complicated cases the property of being self resetting may not be as easy to see. The general procedure is to add the state diagram to itself and see if there are any separate parts, i.e. sets of states from which the desired states cannot be reached, and if there are no separate parts the device is self resetting. Several types of out of sync state are shown in Fig. 10.

If a device is in a correct state and if the signal to it is with-error then it will remain in the correct state. However, a device started correctly can later be in the wrong state by two causes: internal malfunctioning and thru an error in the input sequence. The latter cause can

be examined by a slight variation in the addition process. In Fig 11 a device is shown being fed by two signals, the second seignal being the first as distorted by a noisy channel. The state diagram is first modified by discarding all labels, since the labels are meaningless if the signal digits are possibly in error. Then the resulting diagram is added to itself just as before but imagining that there is only one symbol, ie. "change state". In the example, any state of the sum diagram can be reached from any other, hence all combinations of out of sync conditions can be caused by channel error. That this is not always possible can be seen by considering the non-self resetting diagram of Fig. 9; no channel error can affect the sync of a device when all signals are the same length.

Another way to examine the sync properties of a encoder-decoder combination is to multiply their diagrams according to the rule given above. If the product diagram has a set of states from which the identity part of the diagram cannot be reached, than the decoder will not eventually return to synchronism with the encoder once an error is made. Fig 6 shows again that the transducer added to itself above is self resetting. It will be shown below how a main channel sequence can be found which will definitely reset the decoder in such cases.

Application to Picture Code

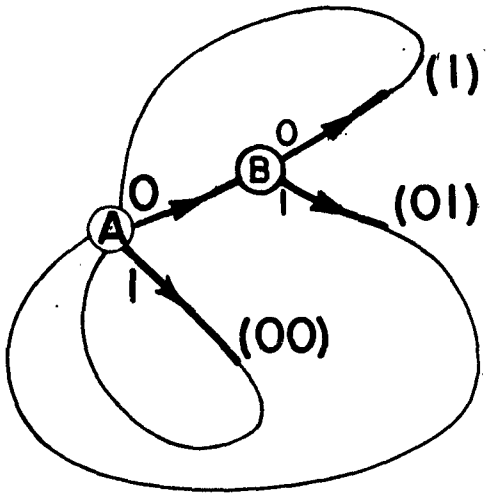
An application of the above methods for examining synchronization can be made to a code which has been used in compressing certain types of facsimile pictures.[3] The code which is to be analyzed is the two component code shown at the bottom of Fig 12. The pictures under consideration are mostly white (0) but where black (1) does occur several black elements can be expected close together. The encoder uses the left component (W) to send white areas and the right component (B) to send black areas. In other words, the system uses two different code tables depending on whether it is sending a white or a black area. When two code tables are used there must be some rule to determine where the transition between them occurs, otherwise the receiver will not know which code table to use. The rule in this case is to always alternate between the two code tables unless the last message in either is encountered (000...00 or 111), in which case the next signal will be chosen from the same code table. The action of the decoding apparatus is to be examined here, and a state diagram representing it is shown at the left in Fig. 12. Since the signals are either 2 or 4 digits long, a simplified diagram can be drawn as shown at the right in Fig. 12 and that the inputs are the 4 combinations of 2 binary digits. The states W and B are, respectively, the base states when the next signal is to be from the W or B code tables. A single pair of binary digits defines the signal in the B table, but 2 pairs are necessary in the W table. As far as sync is concerned, the states marked H', H'' and H''' are similar in that a change is made to the B code table, hence these 3 states can be combined

(3) A.E. Laemmel, "Coding processes for bandwidth reduction in picture transmission", Report R-246-51, Microwave Research Institute, Polytechnic Institute of Brooklyn, August 30, 1951.

into one state (H) as in Fig 13. This is really the operation of reduction as defined above. More amplifications are made in that x is used to represent 01, 10 or 11, and if two arrows go between the same two state they are combined.

The simplified state diagram is now added to itself ("multiplied by 2"), and at the same time another reduction is made by combining pairs of states such as HI and IH. The resulting state diagram, shown on the right in Fig 13, has a set of desired states WW, BB etc. the same as the decoder itself, and also the out of sync states $\overline{HI}$, $\overline{BW}$ etc. Notice first that the desired states can be reached from every out of sync state, but that there are many special sequences which will perpetuate the out of sync condition as long as they are continued. As an example of the latter, start in $\overline{BW}$ and consider the decoder receivers a string of 00's, or start in $\overline{BW}$ and consider a string of x's, say 01011011 ... . The probability of remaining out of sync will usually decrease with increasing time because most sources will not generate the special sequences necessary to maintain the out of sync condition indefinitely.

It is naturally of interest then to ask if there is not some special sequence which will certainly return the decoding apparatus to sync. If it is known what the state of the decorder is this is easily done, for example if the decoder is in state B when it should be in state W, then 11 11 00 would restore it to state B when it should be in state B. But the state of the decoder is not generally known to the transmitter; can a sequence be found which will restore sync regardless of the incorrect state it is in? The answer is again yes, as is shown in the table in Fig. 13. Consider that any of the 6 incorrect state pairs are occuppied, if an x is fed to the decorder then only $\overline{BH}$, $\overline{BW}$ or $\overline{HW}$ can be occuppied. If a 00 is fed to the coder only $\overline{BI}$ can exist, and then a sequence is easily found which reduces this to BB. One such sequence, for the example shown in Fig. 13, is 11 00 11 11 11 00; if this transmitted to the decoder it can be assured that the very next signal will be interpreted in the proper code table. A similar sequence can be found for any decoder (or transducer) if a path exits in the sum of the state diagram with itself from any out of sync state to the desired in sync states. To prove this imagine all out of sync states are occuppied and feed in a signal which removes at least one of them, then remove another etc. Since out of sync states cannot be reached from in sync states, the process must eventually clear all out of sync states.



INPUT		OUTPUT
00	→	1
01	→	01
1	→	00

0		0		0		1		1		0		0		0		0	INPUT
		1				0	1	0	0			1				1	OUTPUT (A)
1				1		0	0	0	0			1				1	OUTPUT (B)

FIG. 1 CODE REPRESENTATION

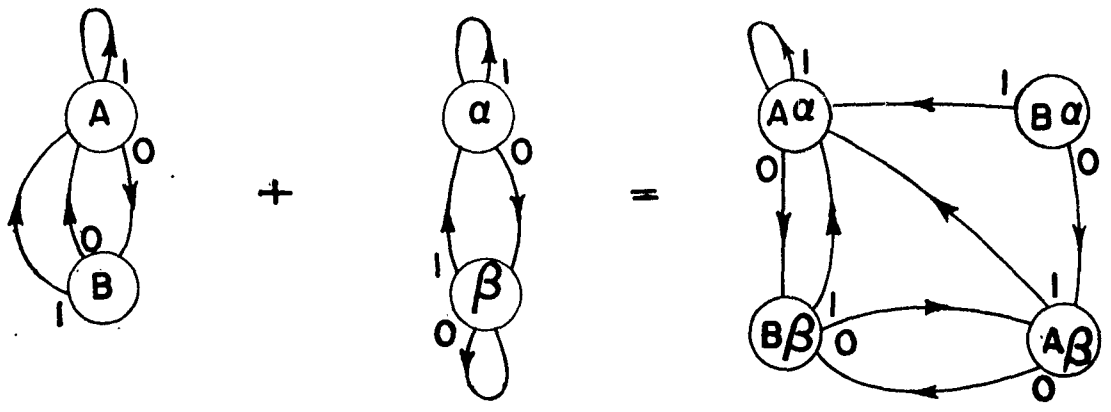


FIG. 2 ADDITION

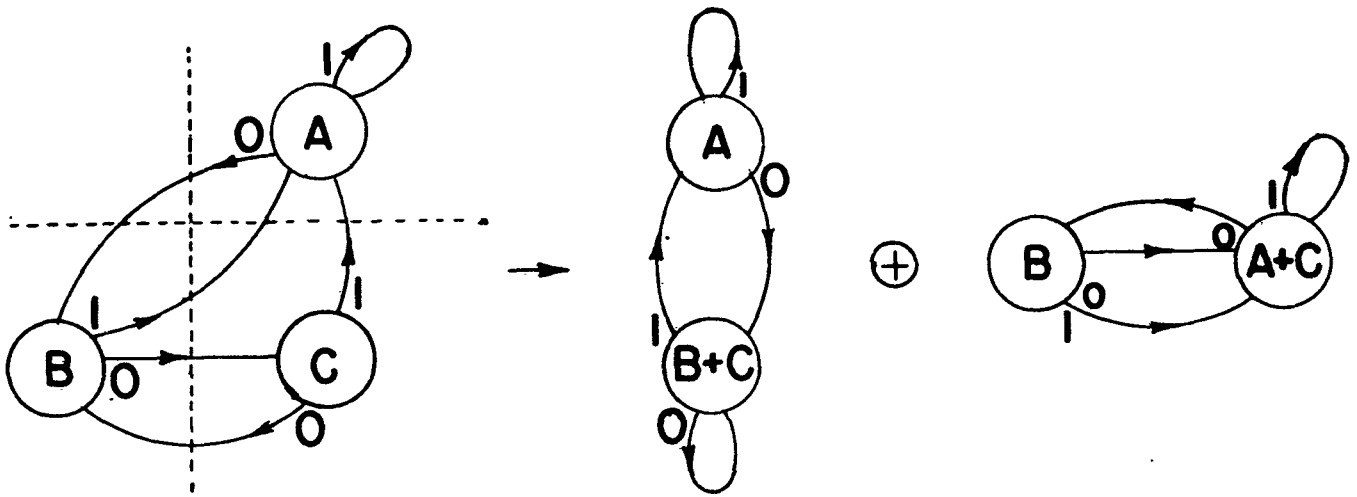


FIG. 3 FACTORIZATION AND REDUCTION

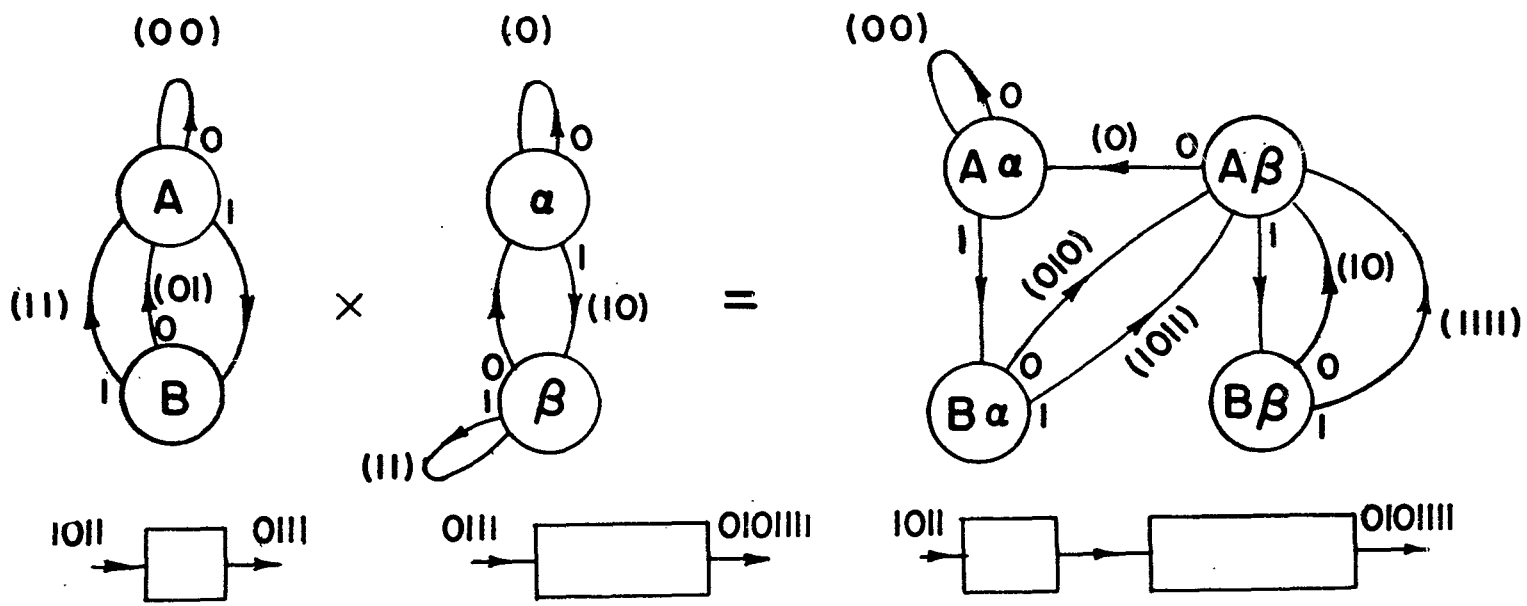
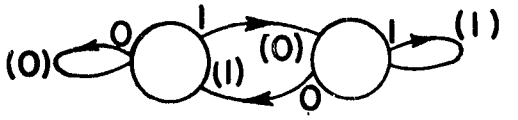


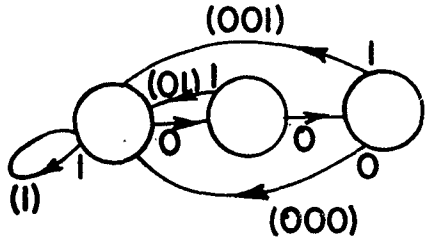
FIG. 4 MULTIPLICATION

M.R.I.-13706

NO DELAY



ONE DIGIT DELAY



VARIABLE DELAY

FIG. 5 IDENTITIES

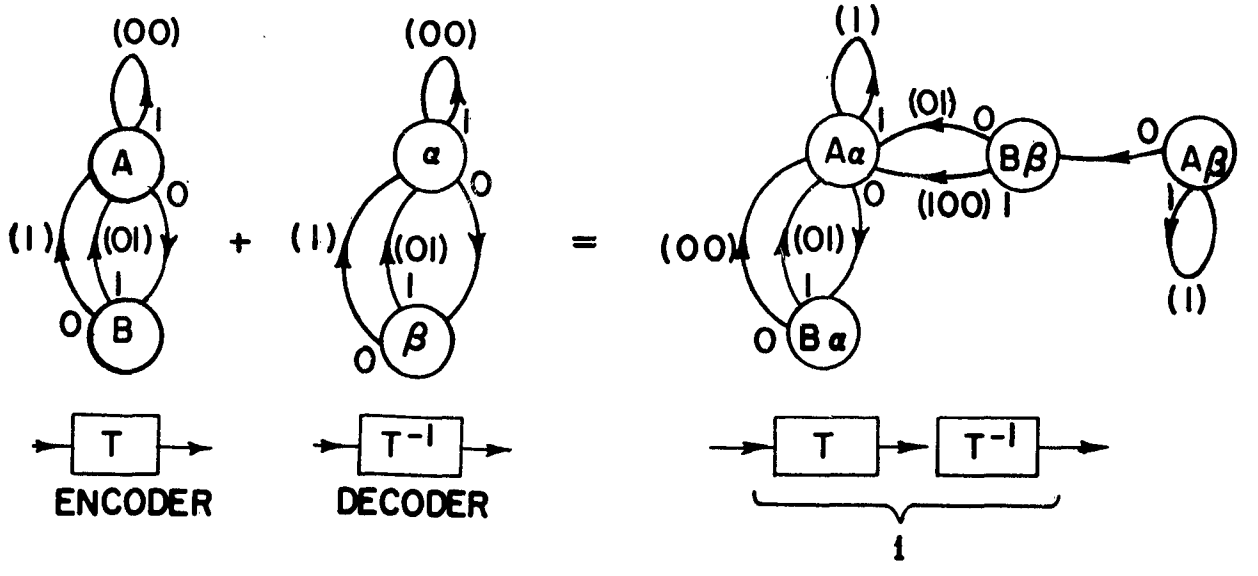


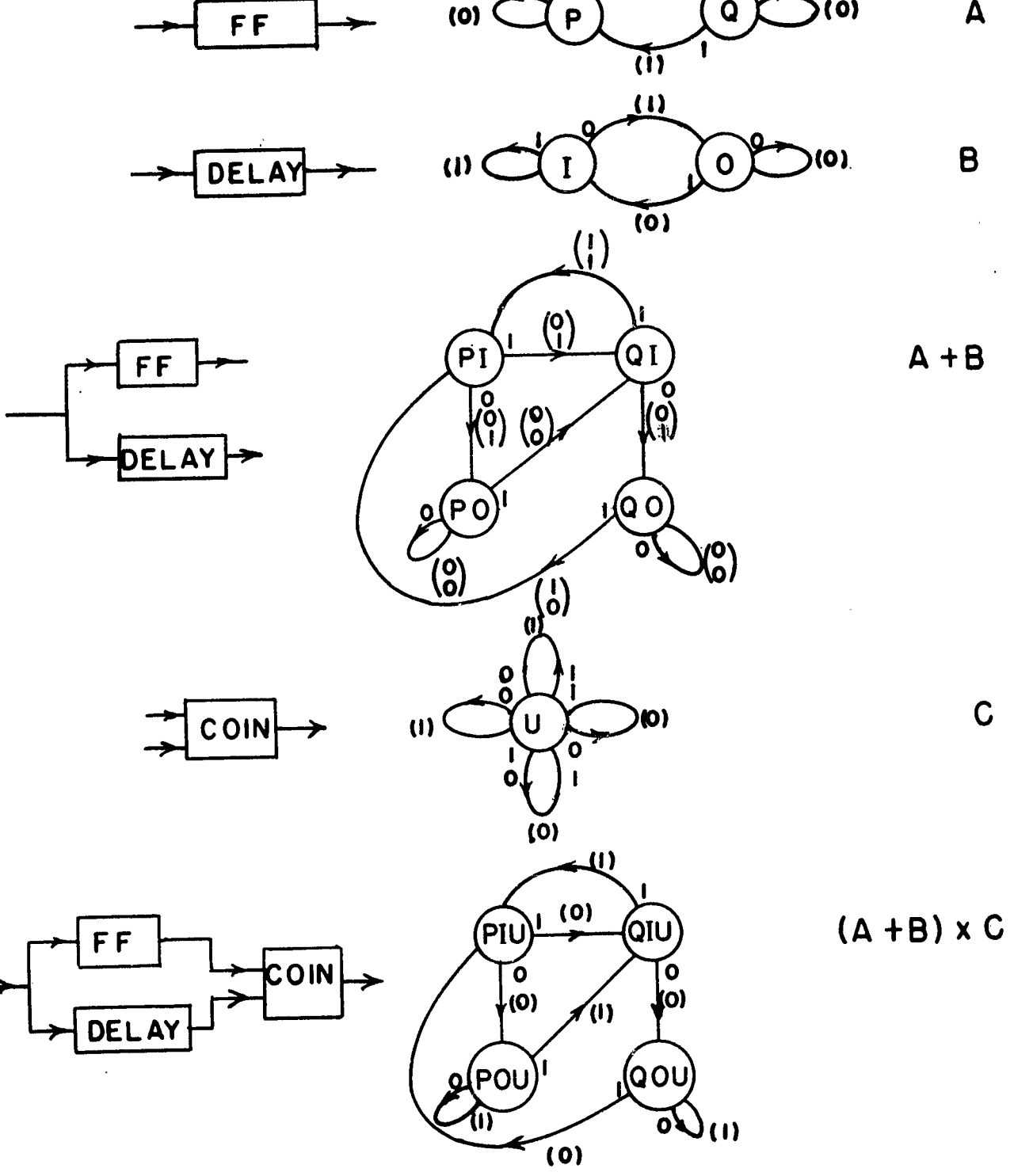
FIG. 6 INVERSE

00 ⟷ 1

01 ⟷ 01

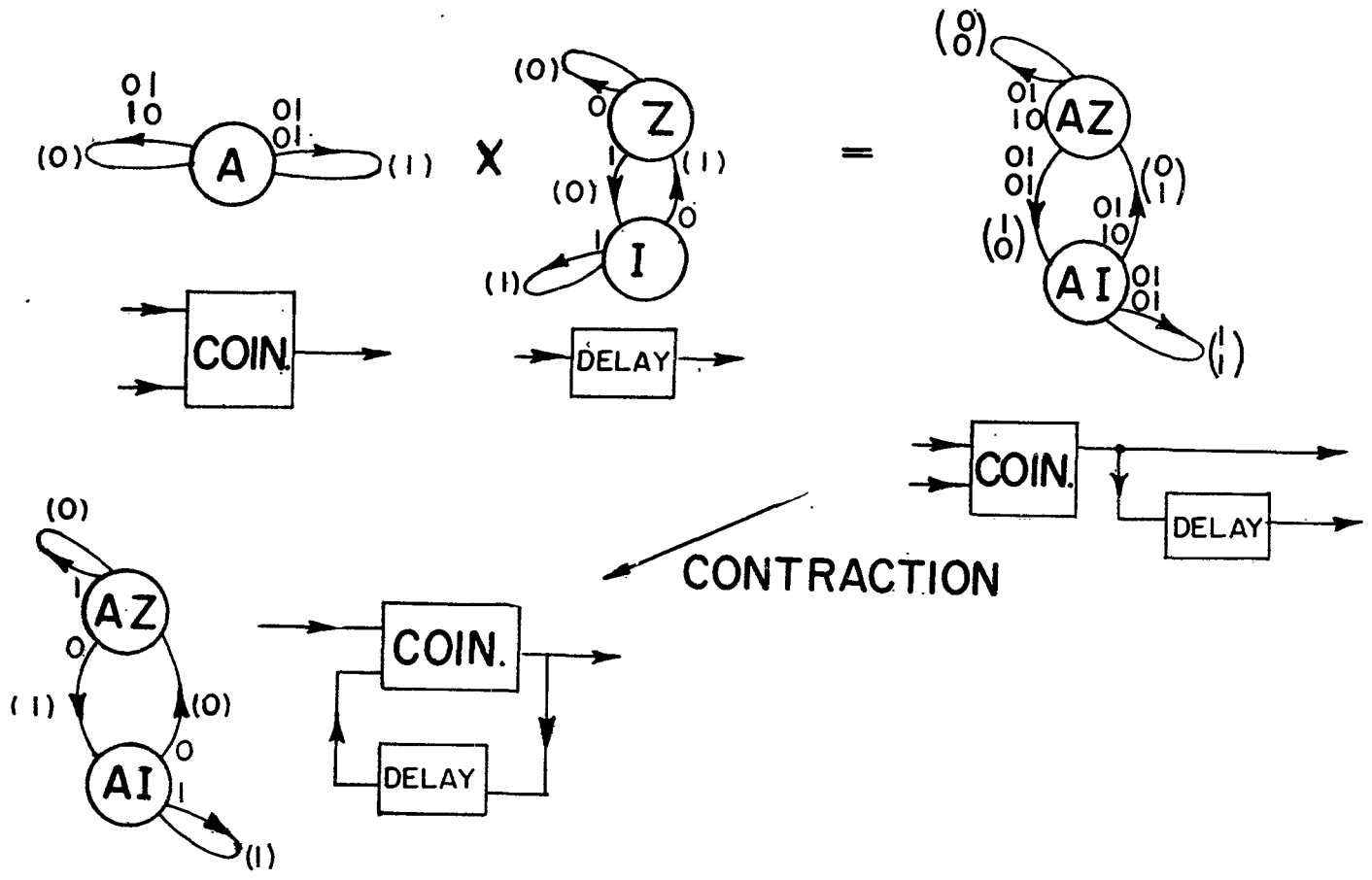
1 ⟷ 00

M.R.I.-13709



MRI-13710

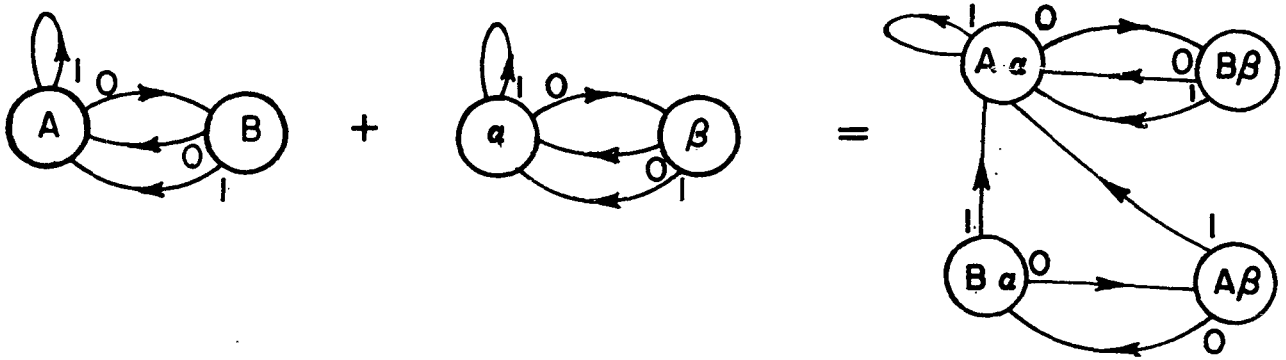
FIG. 7 NETWORK WITHOUT FEEDBACK



FEEDBACK

FIG. 8, M.R.I.-13711

SELF RESETTING



NON SELF RESETTING

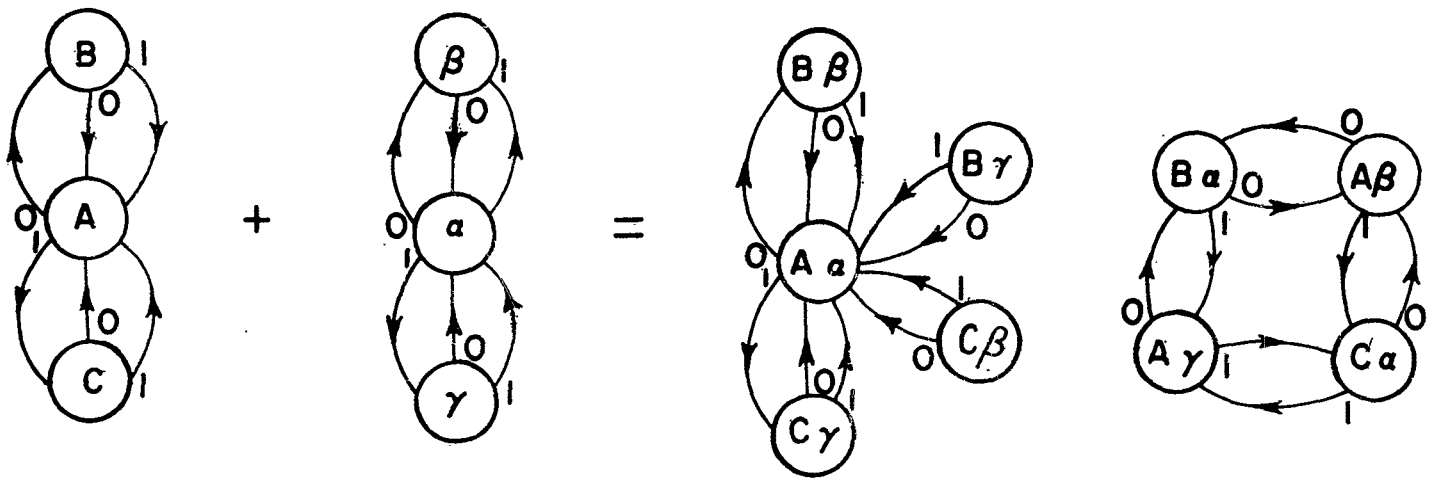


FIG. 9 - M.R.I.-13712

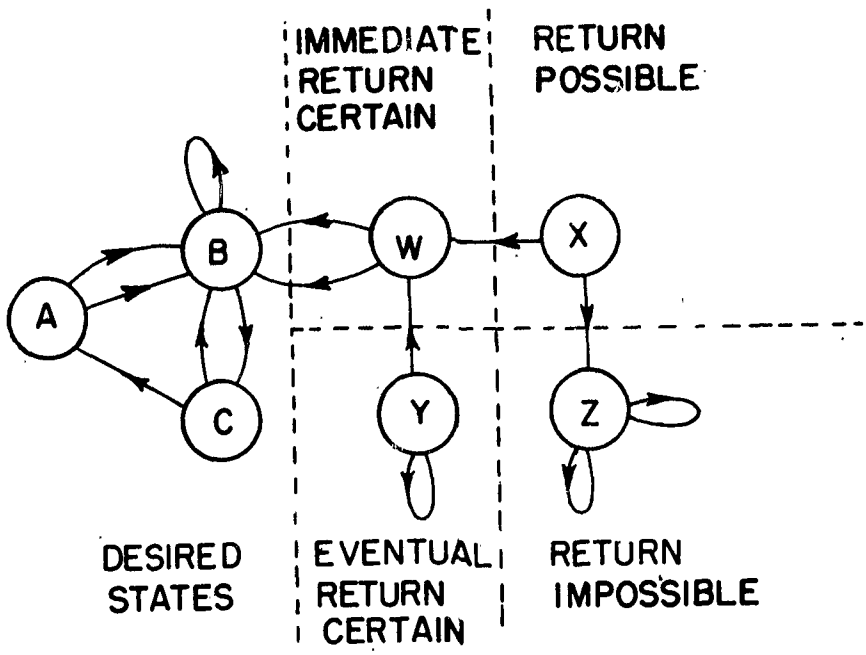


FIG. 10 OUT-OF-SYNC STATES

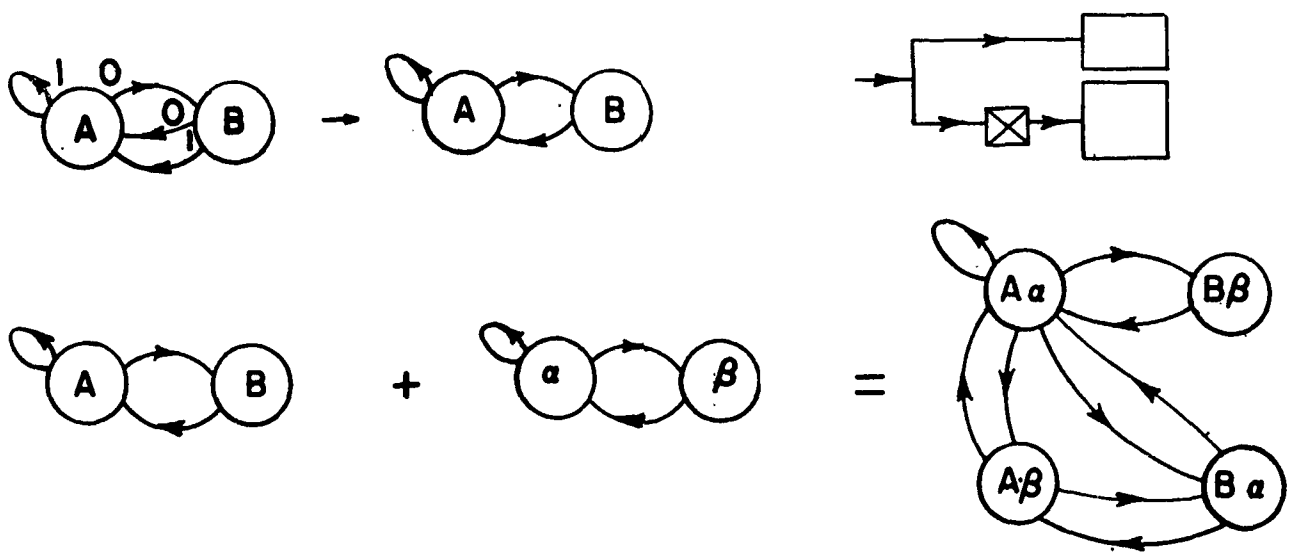


FIG. 11 EFFECT OF INPUT ERRORS

PICTURE CODE

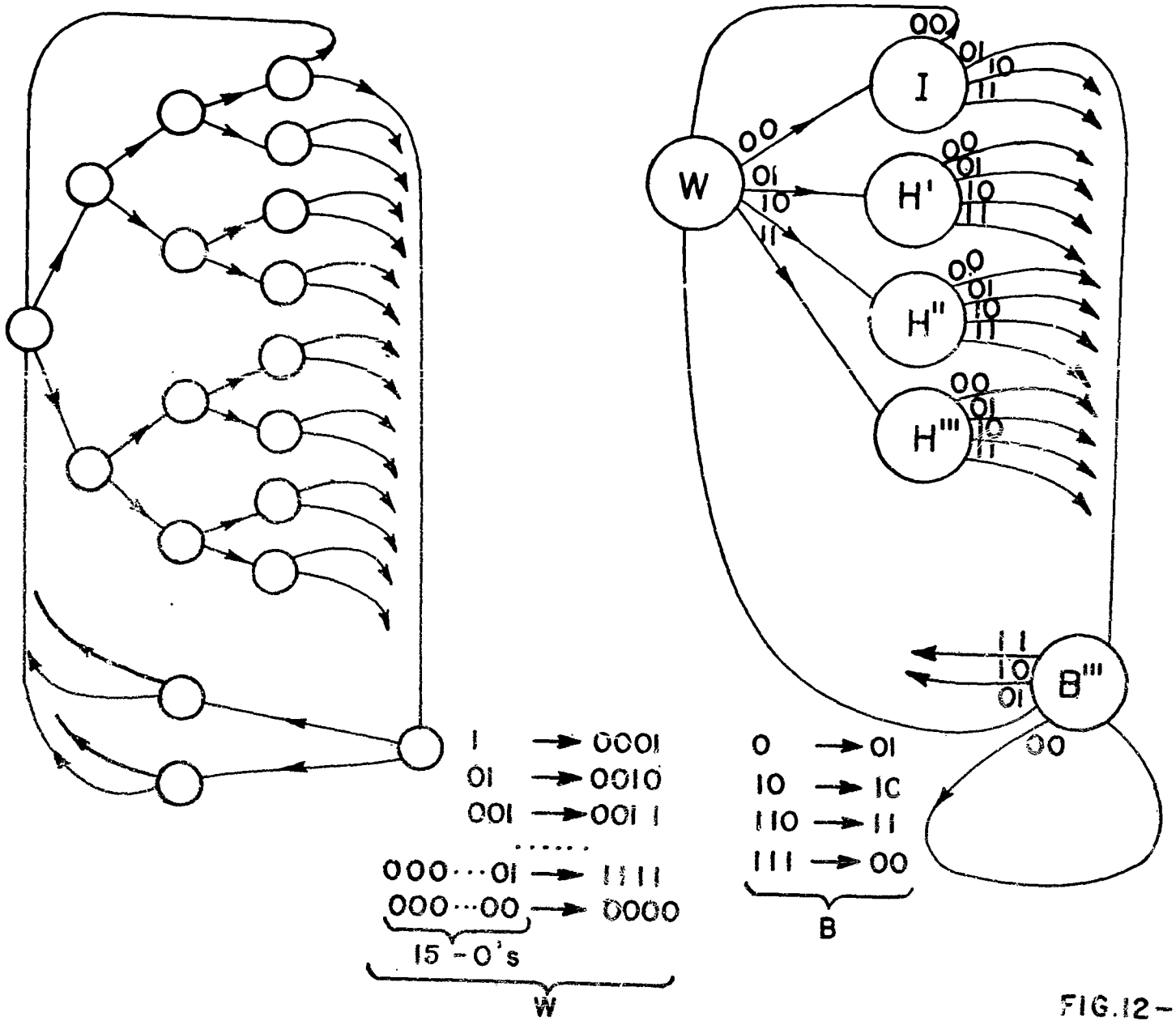
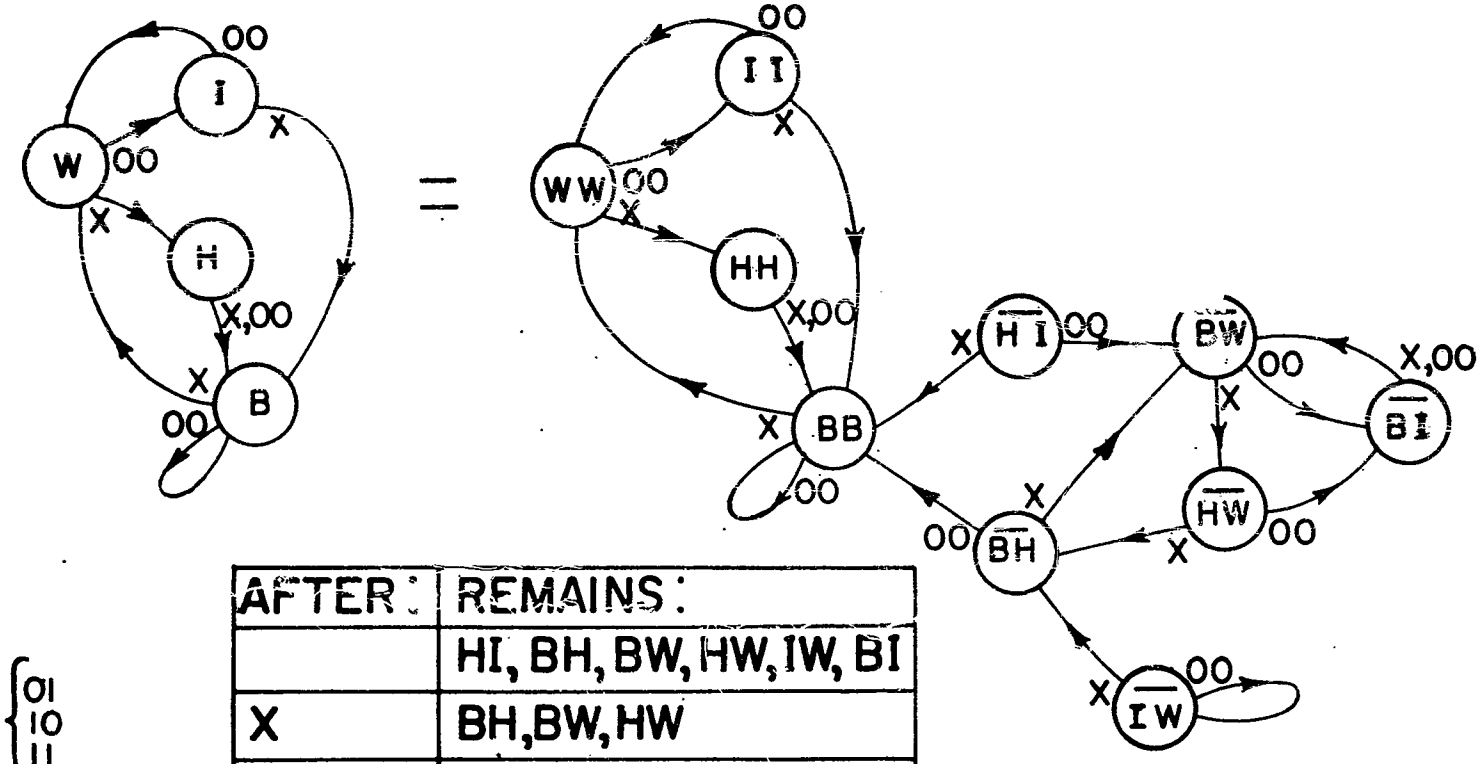


FIG.12-M.R.I.-13714

$\overline{HI} = HI + IH$ etc.

2.



$X = \begin{cases} 01 \\ 10 \\ 11 \end{cases}$

AFTER:	REMAINS:
	HI, BH, BW, HW, IW, BI
X	BH, BW, HW
00	BI
X or 00	BW
X	HW
X	BH
00	NONE

RESET SEQUENCE X00XXX00

FIG. 13 – M.R.I. 13715